LAPORAN PRAKTIKUM DATA STRUCTURE

SIMPLE SORTING

Dosen Pengampu:

H. Fatchurrochman,M.Kom

Asisten Praktikum:

Fillah Anjany 230605110033

Oleh :

Muhammad Alif Mujaddid **240605110082**

Jurusan Teknik Informatika Fakultas Sains dan Teknologi

UIN Maulana Malik Ibrahim Malang

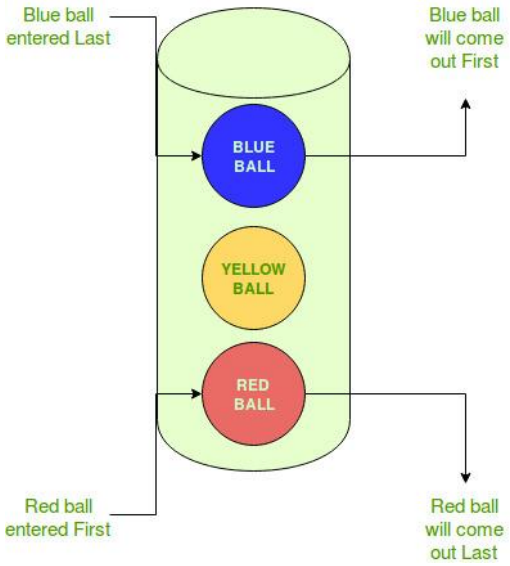
2025

A. PENDAHULUAN

1. Stacks (tumpukan) merupakan suatu susunan koleksi data dimana data yang dapat ditambahkan dan dihapus selalu dilakukan pada bagian akhir data, yang disebut dengan top of stack. Dengan kata lain, stack hanya mengijinkan akses pada item yang terakhir dimasukkan. Stacks bersifat LIFO (Last In First Out). Jelaskan sifat LIFO pada stacks dan gambarkan skema lengkap dari LIFO!

Stack atau tumpukan adalah struktur data yang punya aturan LIFO (Last In, First Out). Artinya, data yang terakhir kali dimasukkan (push) ke dalam stack, akan menjadi data yang pertama kali dikeluarkan (pop).

Analogi paling gampang:
Bayangkan kamu menumpuk piring di dapur. Piring yang terakhir kamu taruh di atas tumpukan adalah piring yang pertama kali bisa diambil. Piring paling bawah hanya bisa diambil kalau semua piring di atasnya sudah diambil dulu.



2. Operasi utama pada Stacks yaitu push dan pop. Selain dua operasi tersebut, juga terdapat operasi peek pada Stacks. Jelaskan masing-masing dari tiga operasi tersebut!

1. Push

Operasi push digunakan untuk menambahkan (menyimpan) data baru ke dalam stack. Data yang ditambahkan akan selalu berada di bagian paling atas (top of stack).
Contoh: stack [A, B, C], kalau di-*push* D hasilnya [A, B, C, D].

2. Pop

Operasi pop digunakan untuk menghapus dan mengambil data dari bagian atas stack. Karena stack bersifat LIFO, data yang terakhir dimasukkan adalah yang pertama kali keluar.
Contoh: stack [A, B, C, D], kalau di-*pop* D keluar, stack jadi [A, B, C].

3. Peek

Operasi peek digunakan untuk melihat data yang ada di paling atas stack tanpa menghapusnya. Jadi, peek hanya memberi tahu isi dari top of stack, tapi stack tetap utuh. Contoh: stack [A, B, C], kalau di-*peek* hasilnya C, tapi stack tetap [A, B, C]

3. Tulislah listing program berikut ini dan jelaskan tiap barisnya!

```
class Stack {


    private int maxSize;
    private long[] stackArray;
    private int top;


    public Stack(int size) {
        maxSize = size;
        stackArray = new long[maxSize];
        top = -1;
    }


    public void push(long item) {
        stackArray[++top] = item;
    }


    public long pop() {
        return stackArray[top--];
    }
```

Membuat class stack

Membuat variable maxSize

Membuat array stackArray

Membuat variable top

Membuat konstruktor untuk stack dan mengisi variablenya

Membuat fungsi push (memasukan data baru di paling atas)

Membuat fungsi pop (menghapus data paling atas)

```
    public long peek() {
        return stackArray[top];
    }

    public boolean isEmpty() {
        return (top == -1);
    }

    public boolean isFull() {
        return (top == maxSize - 1);
    }
}

public class StackApp {

    public static void main(String[] args) {
        Stack theStack = new Stack(10);
        System.out.println(">> push some
                           items");
        theStack.push(20);
        theStack.push(40);
        theStack.push(60);
        theStack.push(80);

        System.out.println("\n>> pop items
                           in the stack");
        while (!theStack.isEmpty()) {
            long value = theStack.pop();
            System.out.print(value + " ");
        }
    }
}
```

Membuat fungsi peek (melihat data paling atas)

Membuat fungsi isEmpty (mebgecek stack kosong atau tidak)

Membuat fungsi isFull(mengecek stack sudah penuh / belum)

Deklerasi class

Membuat method main

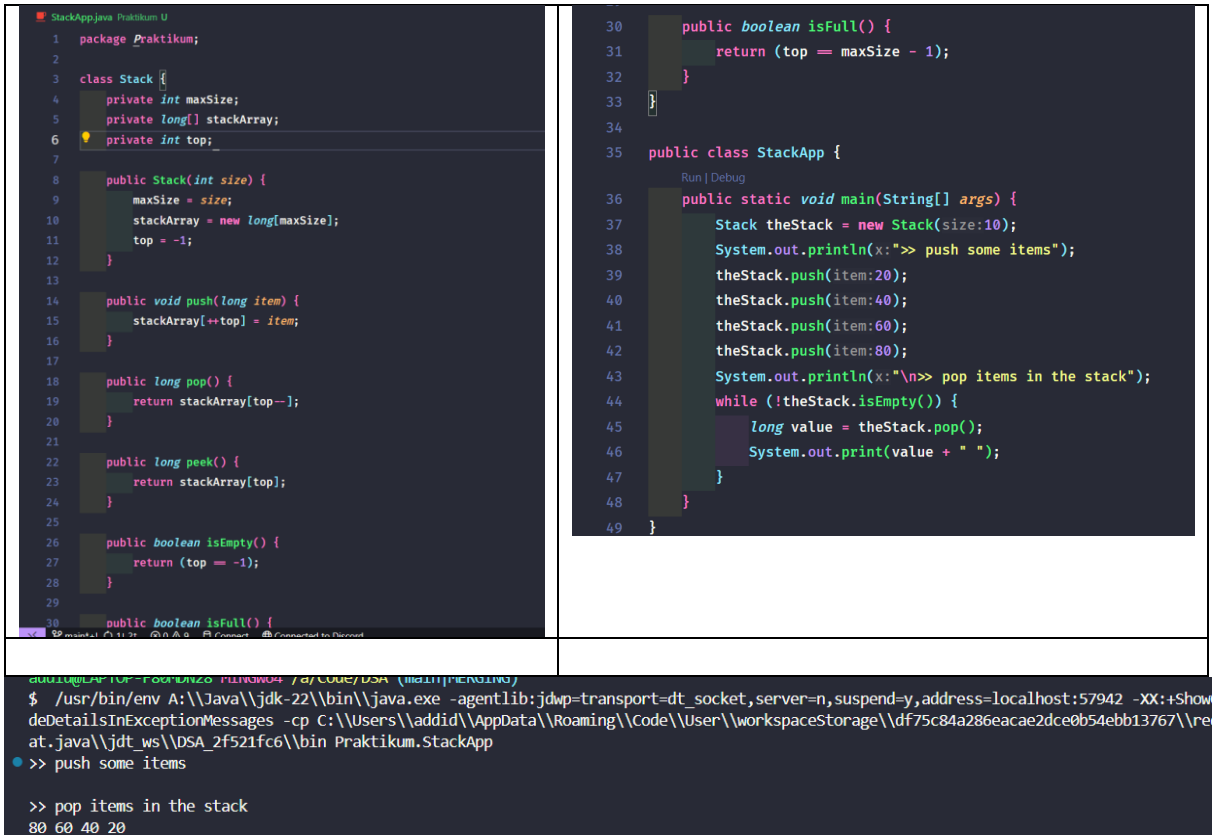
Deklerasi stack dengan max 10

Mengoutputkan “>> push some items”

Menambahkan data

Mengoutputkan “>> pop items in the stack”

Menghapus seluruh data di stack dan mengoutputkanya setiap penghapusan



```
StackApp.java Praktikum U
1  package Praktikum;
2
3  class Stack {
4      private int maxSize;
5      private long[] stackArray;
6      private int top;
7
8      public Stack(int size) {
9          maxSize = size;
10         stackArray = new long[maxSize];
11         top = -1;
12     }
13
14     public void push(long item) {
15         stackArray[++top] = item;
16     }
17
18     public long pop() {
19         return stackArray[top--];
20     }
21
22     public long peek() {
23         return stackArray[top];
24     }
25
26     public boolean isEmpty() {
27         return (top == -1);
28     }
29
30     public boolean isFull() {
31         return (top == maxSize - 1);
32     }
33 }
34
35 public class StackApp {
       Run | Debug
36     public static void main(String[] args) {
37         Stack theStack = new Stack(size:10);
38         System.out.println(x:">> push some items");
39         theStack.push(item:20);
40         theStack.push(item:40);
41         theStack.push(item:60);
42         theStack.push(item:80);
43         System.out.println(x:"\n>> pop items in the stack");
44         while (!theStack.isEmpty()) {
45             long value = theStack.pop();
46             System.out.print(value + " ");
47         }
48     }
49 }
```

```
$  /usr/bin/env A:\\Java\\jdk-22\\bin\\java.exe -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:57942 -XX:+Show
deDetailsInExceptionMessages -cp C:\\Users\\addid\\AppData\\Roaming\\Code\\User\\workspaceStorage\\df75c84a286eacae2dce0b54ebb13767\\re
at.java\\jdt_ws\\DSA_2f521fc6\\bin Praktikum.StackApp
>> push some items

>> pop items in the stack
80 60 40 20
```

Apa output dari listing program tersebut? Jelaskan!

- Baris pertama (>> push some items) muncul karena ada System.out.println.
- Baris kedua (>> pop items in the stack) menunjukkan proses pengeluaran data.
- Angka 80 60 40 20 tercetak sesuai urutan pop, yaitu terbalik dari urutan push, karena stack menggunakan prinsip LIFO (Last In, First Out).
- Dan jika dilihat stack sekarang dalam keadaan empty karena sudah dipop semua

4. Setelah anda memahami tiap baris listing nomer 3. Tuliskan kesimpulan logika untuk setiap method pada class Stack: method push(), pop(), peek(), isEmpty(), dan isFull()!

1. push(long item)

Menambahkan elemen baru ke stack di posisi paling atas. top dinaikkan dulu (++top), lalu item disimpan ke array di indeks tersebut.

2. **pop()**

Mengambil (menghapus) elemen teratas dari stack. Data di indeks top dikembalikan, lalu top diturunkan (top--).

3. **peek()**

Melihat nilai elemen teratas tanpa menghapusnya.Mengembalikan isi dari stackArray[top].

4. **isEmpty()**

Mengecek apakah stack kosong. Jika top == -1, berarti tidak ada elemen dalam stack.

5. **isFull()**

Mengecek apakah stack sudah penuh. Jika top == maxSize - 1, berarti semua slot stack terisi.

5. Salah satu aplikasi yang menggunakan struktur penyimpanan stack adalah parsing ekspresi aritmatika. Tuliskan langkah manual untuk merubah notasi infix: U * (I + N) / M ^ L – G menjadi notasi postfix menggunakan teknik stack!

Token	**Aksi**	**Stack**	**Output**
U	Operand → ke output		U
*	Operator → push ke stack	*	U
(	Push ke stack	* (	U
I	Operand → ke output	* (	U I
+	Operator → push ke stack (di dalam kurung)	* (+	U I
N	Operand → ke output	* (+	U I N
)	Pop sampai (, buang (	*	U I N +
/	Operator → push ke stack (lebih tinggi dari *? sama → pop dulu *)	/	U I N + *
M	Operand → ke output	/	U I N + * M
^	Operator → push ke stack (prioritas tinggi)	/ ^	U I N + * M
L	Operand → ke output	/ ^	U I N + * M L
-	Operator → harus pop ^ (lebih tinggi) dan / (lebih tinggi/equal) → lalu push -	-	U I N + * M L ^ /
G	Operand → ke output	-	U I N + * M L ^ / G

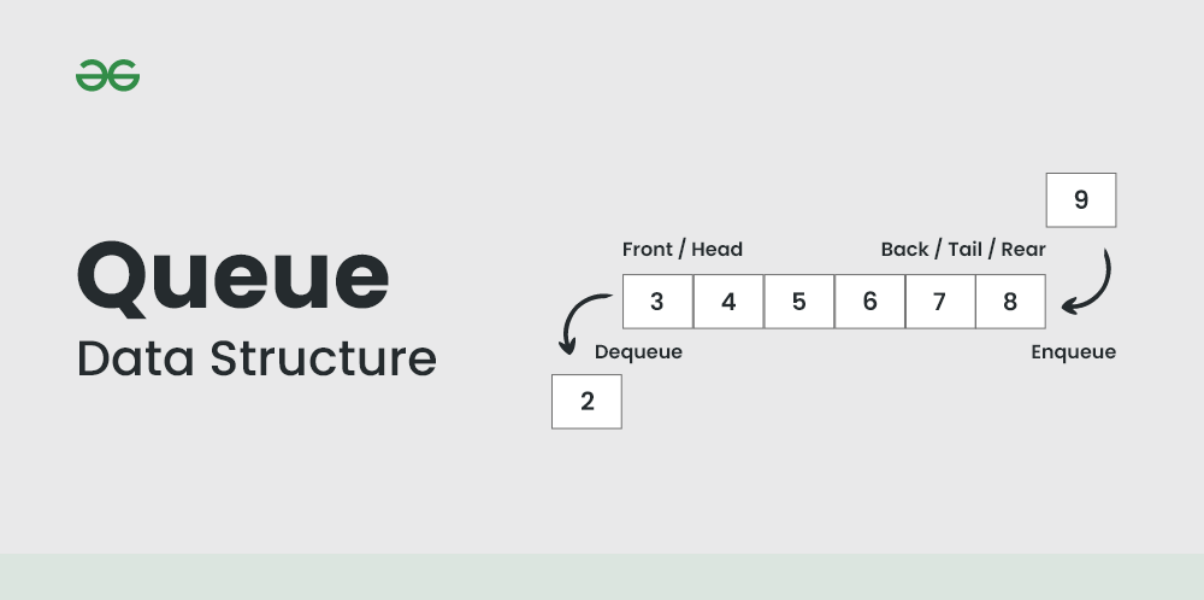
6. Queues (antrian) adalah struktur data yang hampir mirip dengan stack. Perbedaannya adalah pada queues, akses item bagi yang pertama dimasukkan. Queues bersifat FIFO (First In First Out). Jelaskan sifat FIFO pada queues dan gambarkan skema lengkap dari FIFO!

Queue (antrian) bekerja seperti antrian orang di loket/toko.

FIFO (First In, First Out) artinya:

- Elemen yang pertama kali masuk (enqueue) akan keluar lebih dulu (dequeue).
- Elemen yang datang belakangan harus menunggu elemen sebelumnya keluar lebih dulu.

Jadi, urutan keluar sama persis dengan urutan masuk.



7. Berikut ini listing Queue dengan Array. Tulis dan jelaskan!

```
class Queue {
    private int maxSize;
    private long[] queArray;
    private int front;
    private int rear;
    private int nItems;

    public Queue(int size) {
        this.maxSize = size;
        queArray = new long[maxSize];
        front = 0;
        rear = -1;
        nItems = 0;
    }

    public void insert(long value) {
        if (rear == maxSize - 1) {
            rear = -1;
        }
        queArray[++rear] = value;
        nItems++;
    }

    public long remove() {
        long temp = queArray[front++];
        if (front == maxSize) {
            front = 0;
        }
        nItems--;
```

maxSize ukuran maksimum queue. queArray[] array untuk menyimpan data. front indeks elemen paling depan (yang akan keluar duluan).rear indeks elemen paling belakang (tempat masuk data baru). nItems jumlah item di dalam queue.

Membuat konstruktor queue

Membuat fungsi insert untuk memasukan data di belakang rear

Membuat fungsi remove untuk menghapus data dari depan

```
    public long peekFront() {
        return queArray[front];
    }

    public boolean isEmpty() {
        return (nItems == 0);
    }

    public boolean isFull() {
        return (nItems == maxSize);
    }

    public int size() {
        return nItems;
    }
}

public class QueueApp {
    public static void main(String[] args) {
        Queue theQueue = new Queue(5);
        theQueue.insert(10);
        theQueue.insert(20);
        theQueue.insert(30);
        theQueue.insert(40);
        theQueue.remove();
        theQueue.remove();
        theQueue.remove();
        theQueue.insert(50);
        theQueue.insert(60);
        theQueue.insert(70);
        theQueue.insert(80);

        while (!theQueue.isEmpty()){
            long n = theQueue.remove();
            System.out.print(n);
            System.out.print(" ");
        }
        System.out.println("");
    }
}
```

Membuat fungsi peekfront untuk melihat data paling depan

Membuat fungsi isEmpty untuk mengecek apakah kosong

Membuat fungsi isfull untuk mengecek udh penuh belum

Membuat fungsi size untuk mengecek berapa Panjang data

Membuat queue

Memasukan data ke queue

Meremove data paling depan

Memasukan data dari belakang

Perulangan sampe queue 0

Menghapus seluruh data queue dan mengoutputkanya

Tuliskan output dari listing tersebut dan jelaskan!

QueueApp.java Praktikum U

```java
package Praktikum;

class Queue {
    private int maxSize;
    private long[] queArray;
    private int front;
    private int rear;
    private int nItems;

    public Queue(int size) {
        this.maxSize = size;
        queArray = new long[maxSize];
        front = 0;
        rear = -1;
        nItems = 0;
    }

    public void insert(long value) {
        if (rear == maxSize - 1) {
            rear = -1;
        }
        queArray[++rear] = value;
        nItems++;
    }
```

QueueApp.java Praktikum U

```java
class Queue {
    public long remove() {
        if (front == maxSize) {
            front = 0;
        }
        nItems--;
        return temp;
    }

    public long peekFront() {
        return queArray[front];
    }

    public boolean isEmpty() {
        return (nItems == 0);
    }

    public boolean isFull() {
        return (nItems == maxSize);
    }

    public int size() {
        return nItems;
    }
}
```

```java
public class QueueApp {
    Run | Debug
    public static void main(String[] args) {
        Queue theQueue = new Queue(size:5);
        theQueue.insert(value:10);
        theQueue.insert(value:20);
        theQueue.insert(value:30);
        theQueue.insert(value:40);
        theQueue.remove();
        theQueue.remove();
        theQueue.remove();
        theQueue.insert(value:50);
        theQueue.insert(value:60);
        theQueue.insert(value:70);
        theQueue.insert(value:80);
        while (!theQueue.isEmpty()) {
            long n = theQueue.remove();
            System.out.print(n);
            System.out.print(s:" ");
        }
        System.out.println(x:"");
    }
}
```

```
c84a286eacae2dce0b
40 50 60 70 80

addid@LAPTOP-E80MD
```

Queue bekerja dengan prinsip FIFO (First In First Out) data yang pertama dimasukkan adalah yang pertama keluar.

front menunjuk elemen pertama yang akan dihapus, rear menunjuk posisi untuk menambah elemen.

Saat rear mencapai akhir array (maxSize-1), ia kembali ke 0 (circular queue).

Program ini menunjukkan bagaimana data berputar (wrap around) ketika queue penuh tetapi masih ada ruang karena ada data yang sudah di-remove.

8. Setelah anda memahami tiap baris listing nomer 7. Tuliskan kesimpulan logika untuk setiap method pada class Queue: method insert(), remove(), peek(), isEmpty(), dan isFull()!

1. insert(long value)

Logika: Menambahkan elemen baru ke dalam antrian pada posisi rear. Jika rear sudah mencapai akhir array (maxSize-1), maka rear di-wrap around (kembali ke -1, lalu dinaikkan ke 0). Setelah elemen dimasukkan, jumlah item (nItems) bertambah.

2. remove()

Logika: Menghapus elemen dari posisi front (elemen terdepan). front bergerak maju satu posisi. Jika front sudah mencapai akhir array (maxSize), maka di-*wrap around* ke 0. Setelah elemen dihapus, jumlah item (nItems) berkurang.

3. peekFront()

Logika: Mengembalikan (menampilkan) elemen pada posisi front tanpa menghapusnya.

4. isEmpty()

Logika: Mengecek apakah jumlah elemen (nItems) = 0. Jika ya antrian kosong.

5. isFull()

Logika: Mengecek apakah jumlah elemen (nItems) = kapasitas (maxSize). Jika ya antrian penuh.

9. Apa jenis queue (linier/circular) yang diimplementasikan pada listing tersebut? Beri alasan/bukti dari jawaban anda!

queue yang diimplementasikan adalah *circular queue*.

Ada mekanisme wrap around / nilai rear dan front Kembali jika udh nyentuh max